Assignment 2 - Part 1 - CIVENG 263H Fall 2023

Erin jones

Deadline: 9/24/2023

In [28]:
```
import numpy as np
import pandas as pd
import random as rd
import sys, os
import scipy.io as sio
import pickle
rd.seed(50)

# import additional packages

import matplotlib.pyplot as plt #for plots
import matplotlib.cm as cm  #for color maps
import matplotlib as mpl #for color matrices
%matplotlib inline
plt.style.use('fivethirtyeight')
import seaborn as sns # can also be used for plots

from sklearn.decomposition import PCA
from numpy.testing import assert_array_almost_equal

from sklearn.cluster import KMeans
from scipy.spatial.distance import euclidean
```

In [29]:
```
# reads the data and loads data for subject 25
filename = 'realitymining_pick'
infile = open(filename,'rb')
s = pickle.load(infile)
infile.close()

def load_subject_data(subject_id, loaded_pickle):

    subject_data = loaded_pickle['data_mat'][0,(subject_id - 1)].transpose()

    # parse data to corresponding time and days
    hours=np.size(subject_data,1)
    days=np.size(subject_data,0)
    num_labels=5
    subject_array=np.zeros([days,hours*num_labels])
    name_column=[]

    for j in range(1,np.size(subject_array,1)+1):
        J1=hours*(num_labels-1)+1 #97=24*(5-1)+1
        if j >= J1:
            name_column.append(str(j-J1)+'_off')
        else:
            J2=hours*(num_labels-2)+1
            if j>=J2:
                name_column.append(str(j-J2)+'_nsg')
            else:
```

```
                J3=hours*(num_labels-3)+1
                if j >= J3:
                    name_column.append(str(j-J3)+'_els')
                else:
                    J4=hours*(num_labels-4)+1
                    if j >= J4:
                        name_column.append(str(j-J4)+'_wrk')
                    else:
                        name_column.append(str(j-1)+'_hom')

    for i in range(1,days+1):
        for j in range(1,hours+1):
            place=subject_data[i-1][j-1]
            if np.isnan(place):
                Ji=hours*(num_labels-1)+1 #97=24*(5-1)+1
                Jf=hours*num_labels #120
                J=Ji+j-1
                subject_array[i-1][J-1]=1
            else:
                if place==0:
                    Ji=hours*(num_labels-2)+1 #73
                    Jf=hours*(num_labels-1)   #96
                    J=Ji+j-1
                    subject_array[i-1][J-1]=1
                else:
                    Ji=int(hours*(place-1)+1) # (1-24) for house and (25-48) fo
                    Jf=int(hours*place)       # and (49-72) for elsewehere
                    J=int(Ji+j-1)
                    subject_array[i-1][J-1]=1

    return subject_array, hours, days
```

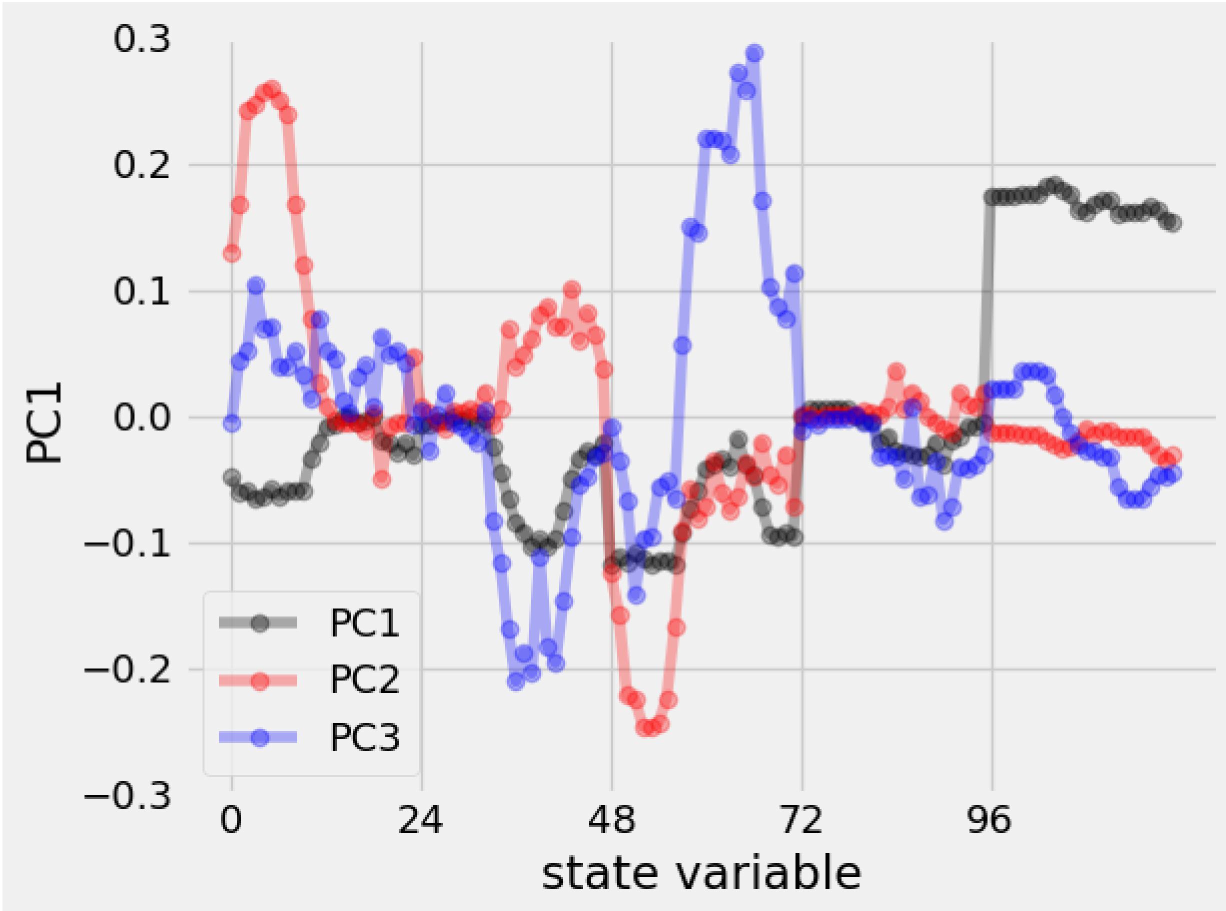
In [30]:

```
Mbw25, hours, days = load_subject_data(25, s)
```

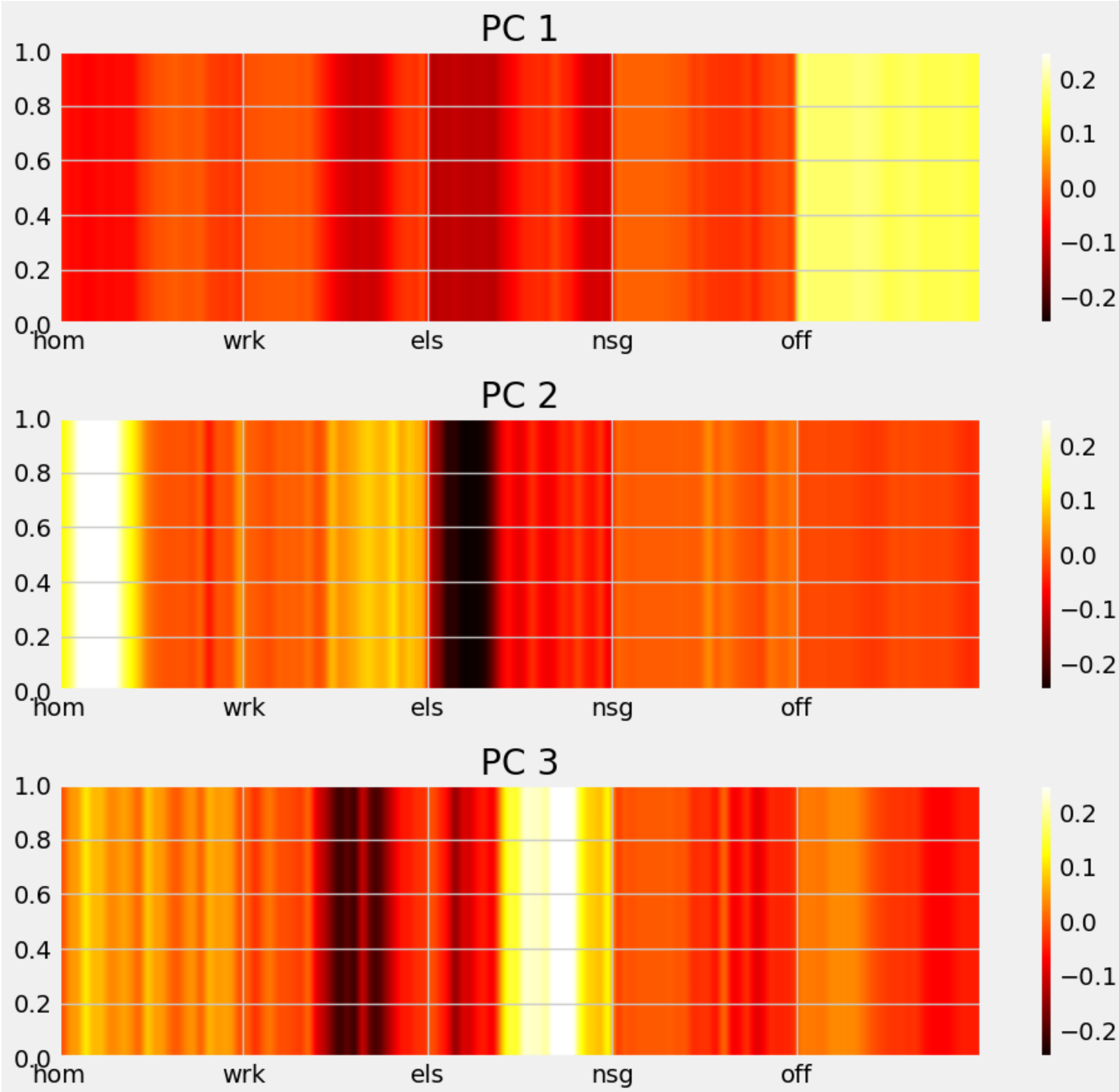
In [55]:

```
# considering all columns as number_c
number_c=120
pca25 = PCA(n_components=number_c)  ##Estimates the components
pca25.fit(Mbw25)
Mbw_pca25=pca25.fit_transform(Mbw25)
Mbw_recons25 = pca25.inverse_transform(Mbw25)

# shows the plot of the first three PCs
x = np.arange(hours*num_labels)
plt.plot(x,pca25.components_[0,:], 'o-', color='black', alpha=0.3, label= "PC1'
plt.plot(x,pca25.components_[1,:], 'o-', color='red', alpha=0.3, label= "PC2")
plt.plot(x,pca25.components_[2,:], 'o-', color='blue', alpha=0.3, label= "PC3")
plt.ylim([-0.3, 0.3])
plt.xticks(np.arange(0, 120, step=24))
plt.xlabel("state variable")
plt.ylabel("PC1")
plt.tight_layout()
plt.legend()
plt.savefig("./11pc1.pdf")
plt.show()


# reproduces the figure from the paper

PC_dictionnary = {} #this step creates a Data frames with the PCs
```

```
Principal_components_names = ['PC ' + str(i) for i in range(1,len(name_column)+
for idx in range(len(Principal_components_names)):
    PC_dictionnary[Principal_components_names[idx]] = pca25.components_[idx]

PC_data = pd.DataFrame(data = PC_dictionnary)
print("The principal components of this dataset are:")
print(PC_data)

PC_np = PC_data.values
name_state = ['hom','wrk','els','nsg','off']
fig, (ax1,ax2,ax3) = plt.subplots(nrows=3, figsize=(10,9))

im=ax1.imshow(np.transpose(PC_np[:,0:1]),vmin=-0.25,vmax=0.25,extent=[0,120,0,1
ax1.set_title('PC 1')
fig.colorbar(im, ax=ax1)
ax1.set_xticks(np.arange(0,120,24))
ax1.set_xticklabels(name_state)

im2=ax2.imshow(np.transpose(PC_np[:,1:2]),vmin=-0.25,vmax=0.25,extent=[0,120,0,
ax2.set_title('PC 2')
fig.colorbar(im2, ax=ax2)
ax2.set_xticks(np.arange(0,120,24))
ax2.set_xticklabels(name_state)

im3=ax3.imshow(np.transpose(PC_np[:,2:3]),vmin=-0.25,vmax=0.25,extent=[0,120,0,
ax3.set_title('PC 3')
fig.colorbar(im3, ax=ax3)
ax3.set_xticks(np.arange(0,120,24))
ax3.set_xticklabels(name_state)

plt.tight_layout()
# plt.savefig("./11pc2.pdf")
```


0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
PC1
0
24
48
72
96
state variable
PC1
PC2
PC3

```
The principal components of this dataset are:
         PC 1      PC 2      PC 3      PC 4      PC 5      PC 6      PC 7  \
0   -0.049030  0.129238 -0.005709  0.105796  0.050763  0.049884 -0.122795
1   -0.060461  0.168186  0.043461  0.072531  0.098697 -0.046446 -0.120924
2   -0.059949  0.242737  0.051737  0.094076  0.029519 -0.078249 -0.055873
3   -0.066492  0.246476  0.103658  0.077253 -0.022807 -0.079747 -0.021423
4   -0.064836  0.255817  0.069745  0.093419 -0.018141 -0.096496  0.023964
..        ...       ...       ...       ...       ...       ...       ...
115  0.161582 -0.015955 -0.066333  0.107855 -0.029723 -0.142212 -0.029948
116  0.165374 -0.022317 -0.056542  0.110566 -0.041170 -0.134022 -0.040836
117  0.162645 -0.030791 -0.046598  0.131420 -0.046492 -0.110156 -0.043513
118  0.154894 -0.035659 -0.048302  0.138650 -0.043386 -0.113321 -0.049851
119  0.153927 -0.031637 -0.045855  0.147634 -0.041013 -0.111558 -0.059162

         PC 8      PC 9     PC 10  ...    PC 111        PC 112        PC 113
\
0    0.089900  0.191870  0.243763  ... -0.000000  0.000000e+00 -0.000000e+00
1    0.105810  0.303604 -0.008712  ...  0.027527  4.326734e-16 -2.389738e-18
2    0.033330 -0.017745 -0.040384  ...  0.103733  3.997271e-15  4.690364e-17
3    0.031697  0.008598 -0.055180  ... -0.007746 -3.916714e-15  3.518058e-17
4    0.048329 -0.087358  0.029894  ... -0.130750 -4.009876e-15  8.510449e-17
..        ...       ...       ...  ...       ...           ...           ...
115 -0.039251  0.006314  0.044611  ... -0.005859  7.731143e-17  1.347322e-17
116 -0.027929 -0.012204  0.051357  ... -0.046054 -1.925874e-15  4.117153e-17
117  0.001869 -0.041196  0.045470  ...  0.150118  4.772137e-15 -3.330216e-17
118  0.006912 -0.062871  0.068659  ...  0.046681  3.759443e-15 -9.096432e-18
119  0.033594 -0.074202  0.066452  ... -0.075811 -4.119290e-15  6.361055e-17

           PC 114        PC 115        PC 116        PC 117        PC 118  \
0   -0.000000e+00 -0.000000e+00 -0.000000e+00  0.000000e+00 -0.000000e+00
1   -2.008739e-17  1.961286e-18  2.723305e-17 -1.307766e-17 -7.086993e-19
2   -2.531108e-17  1.509556e-17  2.323192e-17  2.383442e-17  1.684983e-18
3   -5.462747e-17  2.742392e-17  4.823559e-17  5.111753e-17 -3.054642e-18
4   -2.266730e-17 -6.506906e-17  9.658737e-17 -6.005332e-18 -5.330389e-17
..            ...           ...           ...           ...           ...
115 -2.386749e-16  5.359147e-16  6.348637e-16 -3.022619e-16 -6.150411e-01
116 -4.820087e-17 -2.050785e-17  4.461905e-17  6.316010e-17 -5.155175e-17
117 -5.443745e-18  8.347277e-18 -7.825841e-18 -4.870097e-17  7.052938e-17
118 -4.781897e-17  2.112848e-18 -5.149273e-18  2.613699e-18 -8.048286e-17
119 -1.110246e-16  1.066142e-16  6.398894e-17  8.830556e-17 -5.140297e-17

           PC 119    PC 120
0    0.000000e+00 -0.000000
1    7.261225e-18 -0.190301
2    4.057730e-17 -0.276833
3   -2.444968e-17 -0.010967
4   -2.065441e-17 -0.064150
..            ...       ...
115  5.370205e-01 -0.013240
116 -2.917597e-17  0.076484
117 -1.951552e-17  0.165491
118 -6.607394e-17 -0.057756
119 -2.311665e-17  0.064788

[120 rows x 120 columns]
```



Question 1

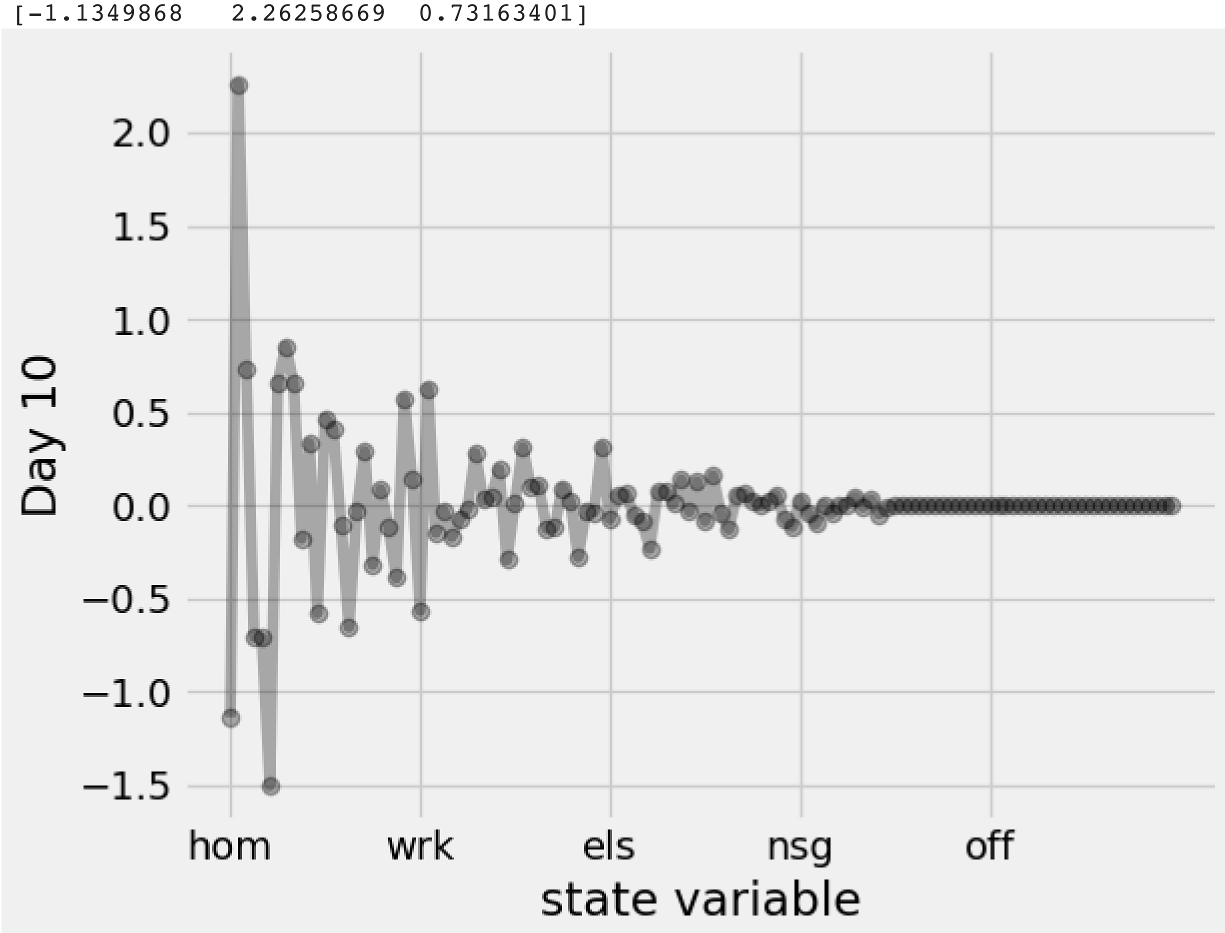
Based on the plot above, PC1 does a great job at representing the eigenbehavior where the user has their device turned off for essentially the entire day. PC2 does a fantastic job assessing when someone is home or elsewhere in the first half of the day (strong positive/negative projections). PC3 does an excellent job describing a behavior where the subject is elsewhere for the latter 2/3 of the day (possibly running errands?) and describes the occassional behavior where there is a stop at work in the afternoon/evening. There are also hints of the individual active at home for bits and pieces of the day, but nothing when compared to the magnitude of the PC3 value for the elsewhere category. See the description of the activity for each day below in the markdown below the given plot.

In [32]:
```
Mbw_pca25.shape
```

Out[32]:
```
(162, 120)
```

In [62]:
```
# Day 10
x = np.arange(hours*num_labels)
plt.plot(x,Mbw_pca25[9,:], 'o-', color='black', alpha=0.3)
# plt.xticks(np.arange(0, 120, step=24))
plt.xticks(np.arange(0, 120, step=24), labels=name_state)
plt.xlabel("state variable")
plt.ylabel("Day 10")
plt.tight_layout()

# prints the tenth day and 3 projections (Principal components for specific day
print("The 3 projections for Day 10 are: ")
print(Mbw_pca25[9,:3])
```

```
The 3 projections for Day 10 are:
[-1.1349868   2.26258669  0.73163401]
```

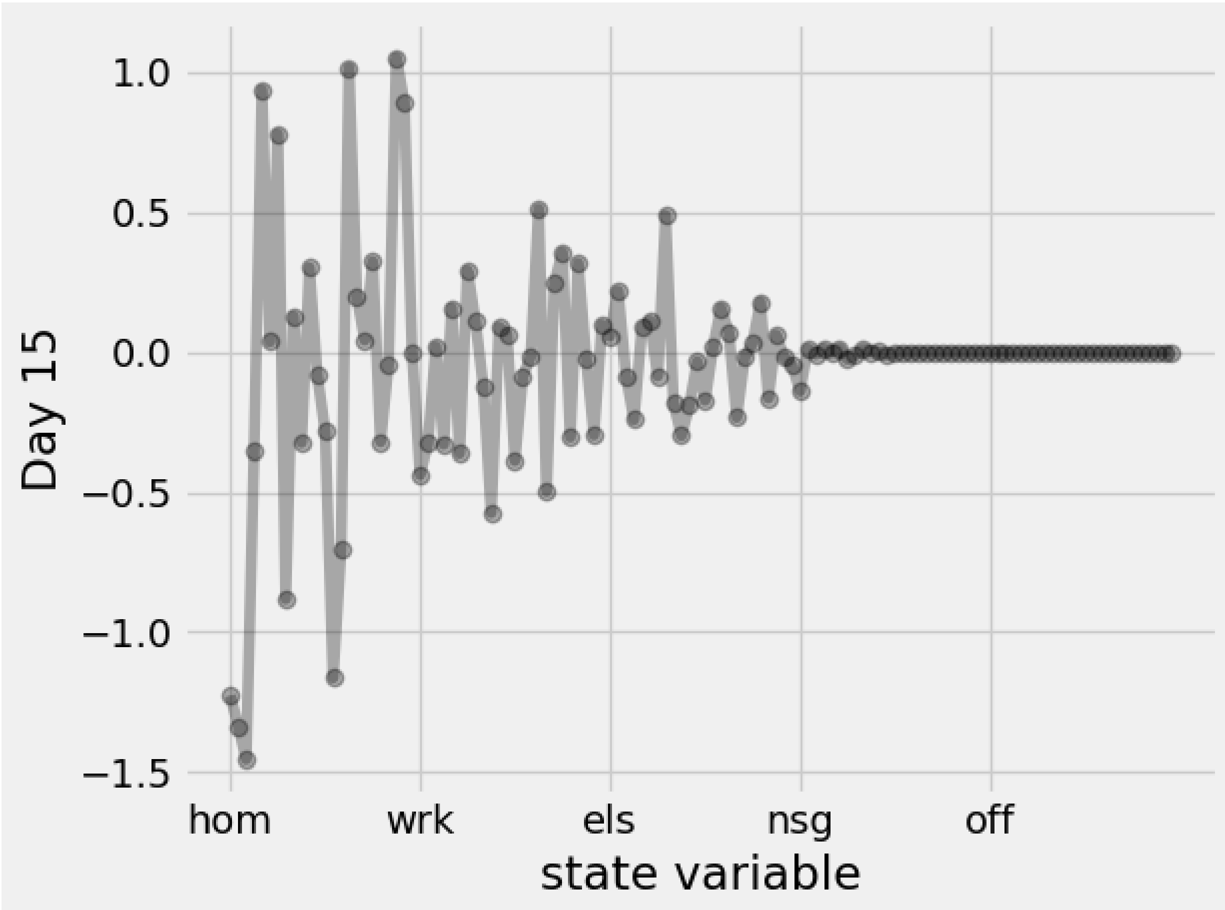


The pattern for day 10 appears to represent very high likelihood that the individual is home during the first part of their day and then less so as the day goes on. There also appears to be a good amount of activity during the middle of the day related to their being at work. Day 10 has little activity elsewhere (maybe a weekday) and it definitely seems that their device was not likely to be found off on that day.

In [58]:
```
# Day 15
x = np.arange(hours*num_labels)
plt.plot(x,Mbw_pca25[14,:], 'o-', color='black', alpha=0.3)
plt.xticks(np.arange(0, 120, step=24), labels=name_state)
plt.xlabel("state variable")
plt.ylabel("Day 15")
plt.tight_layout()
```

```
# prints the first day and 3 projections (Principal components for specific day
print("The 3 projections for Day 15 are: ")
print(Mbw_pca25[14,:3])
```

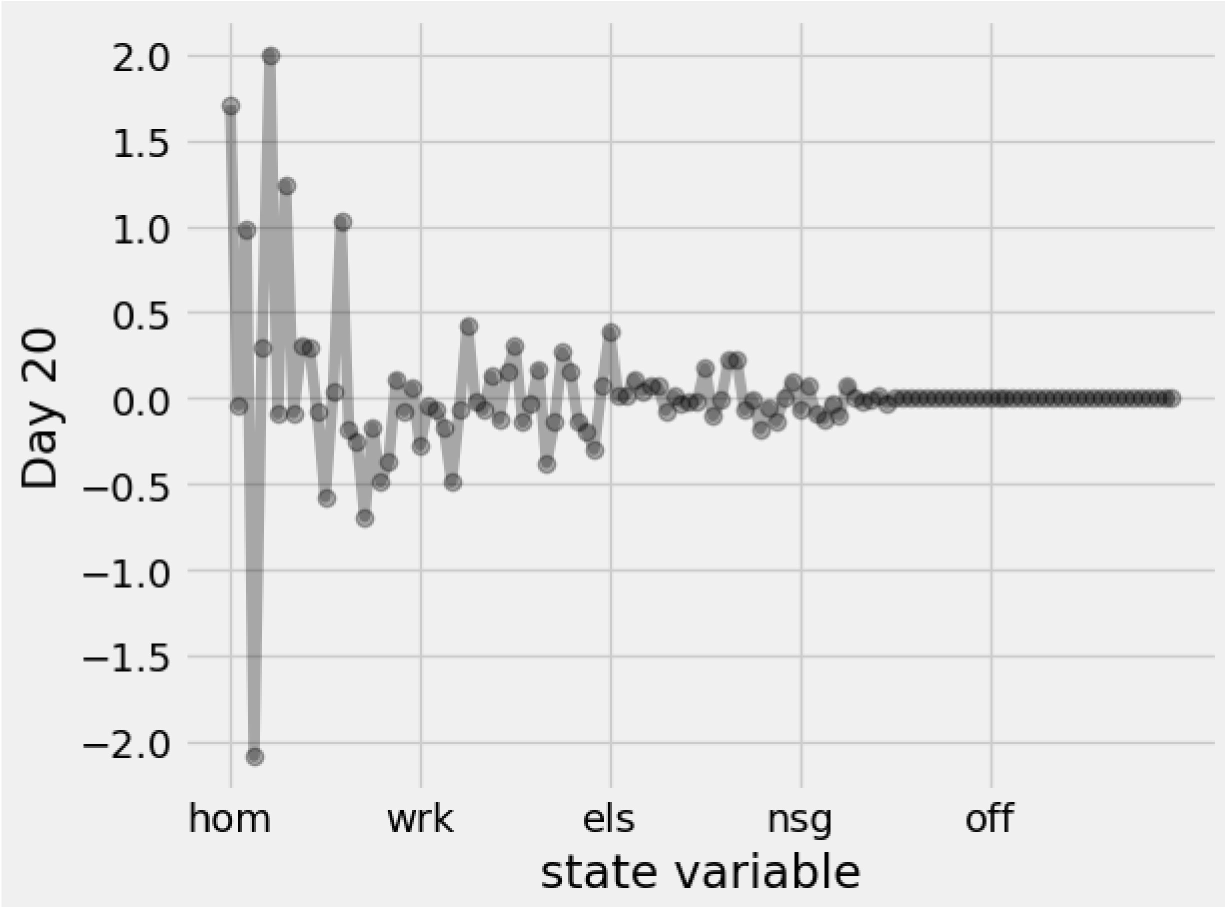
```
The 3 projections for Day 15 are:
[-1.22729175 -1.33670925 -1.45511471]
```



On day 15, the subject spent a good amount of time going to and from home, with an equal amount of time appearing to be spent either elsewhere or at work. This is a day not spent straight between home and work, but rather maybe a day that was also spent elsewhere. It seems their device was on the majority of the time.

In [59]:
```
# Day 20
x = np.arange(hours*num_labels)
plt.plot(x,Mbw_pca25[19,:], 'o-', color='black', alpha=0.3)
plt.xticks(np.arange(0, 120, step=24), labels=name_state)
plt.xlabel("state variable")
plt.ylabel("Day 20")
plt.tight_layout()

# prints the first day and 3 projections (Principal components for specific day
print("The 3 projections for Day 20 are: ")
print(Mbw_pca25[19,:3])
```

```
The 3 projections for Day 20 are:
[ 1.70321062 -0.04325883  0.97991906]
```



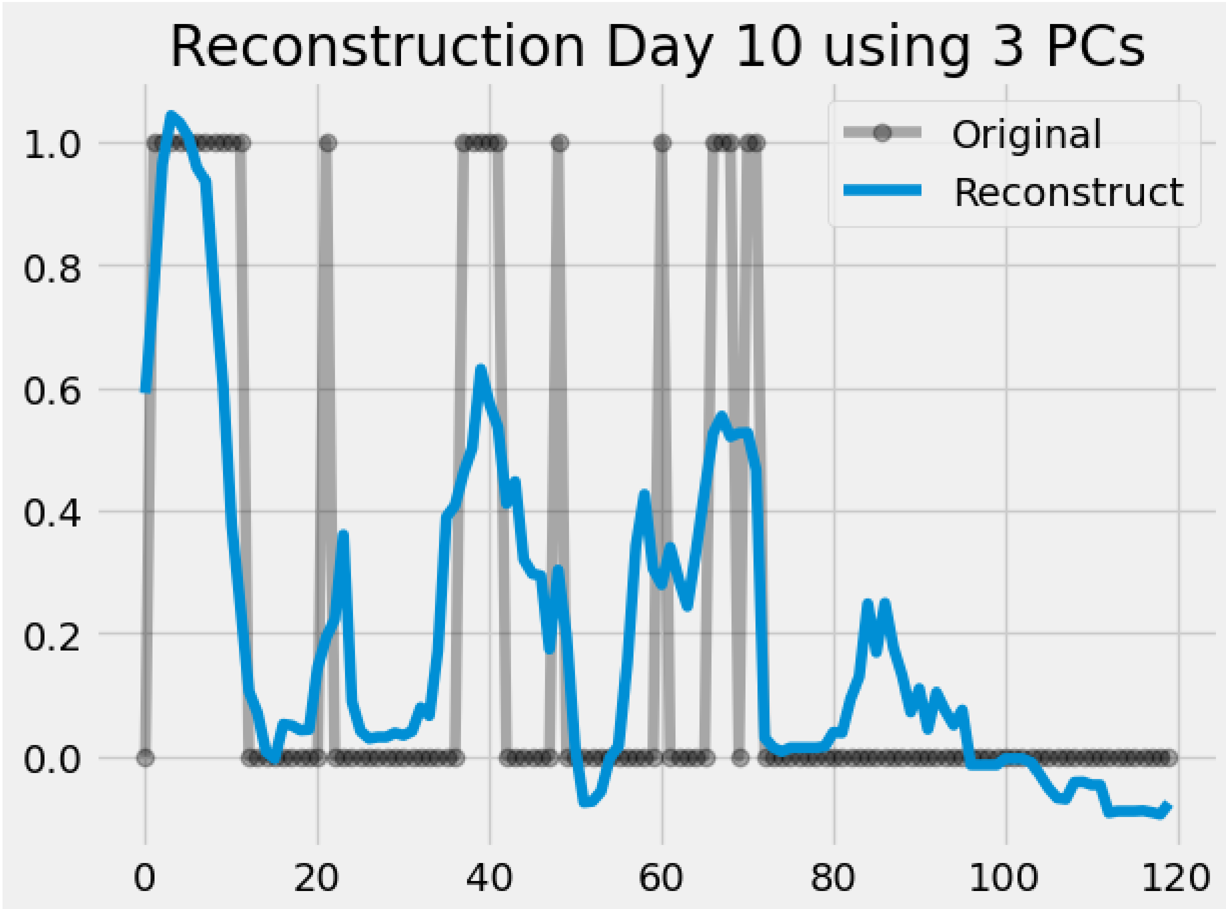
Day 20 is very similar in terms of behavior patterns to Day 10, however more time is spent at home than on that day. There also seems to be less activity elsewhere than on other days. Again, this subject appears to like to have their phone on a whole lot!

Question 2

In [36]:
```
# Reconstruction of Sample Days with First 3 Eigenvectors

pca25_3 = PCA(3)

Mbw25_pca3= pca25_3.fit_transform(Mbw25)
MBW_recons25_3 = pca25_3.inverse_transform(pca25_3.fit_transform(Mbw25))
```

In [37]:
```
# shows the reconstruction of Day 10

x = np.arange(hours*num_labels)
plt.plot(x,Mbw25[9,:], 'o-', color='black', alpha=0.3,label="Original")
plt.plot(x,MBW_recons25_3[9,:],label="Reconstruct")

plt.title('Reconstruction Day 10 using 3 PCs')
plt.tight_layout()
plt.legend()
```

Out[37]:
```
<matplotlib.legend.Legend at 0x30937c3d0>
```

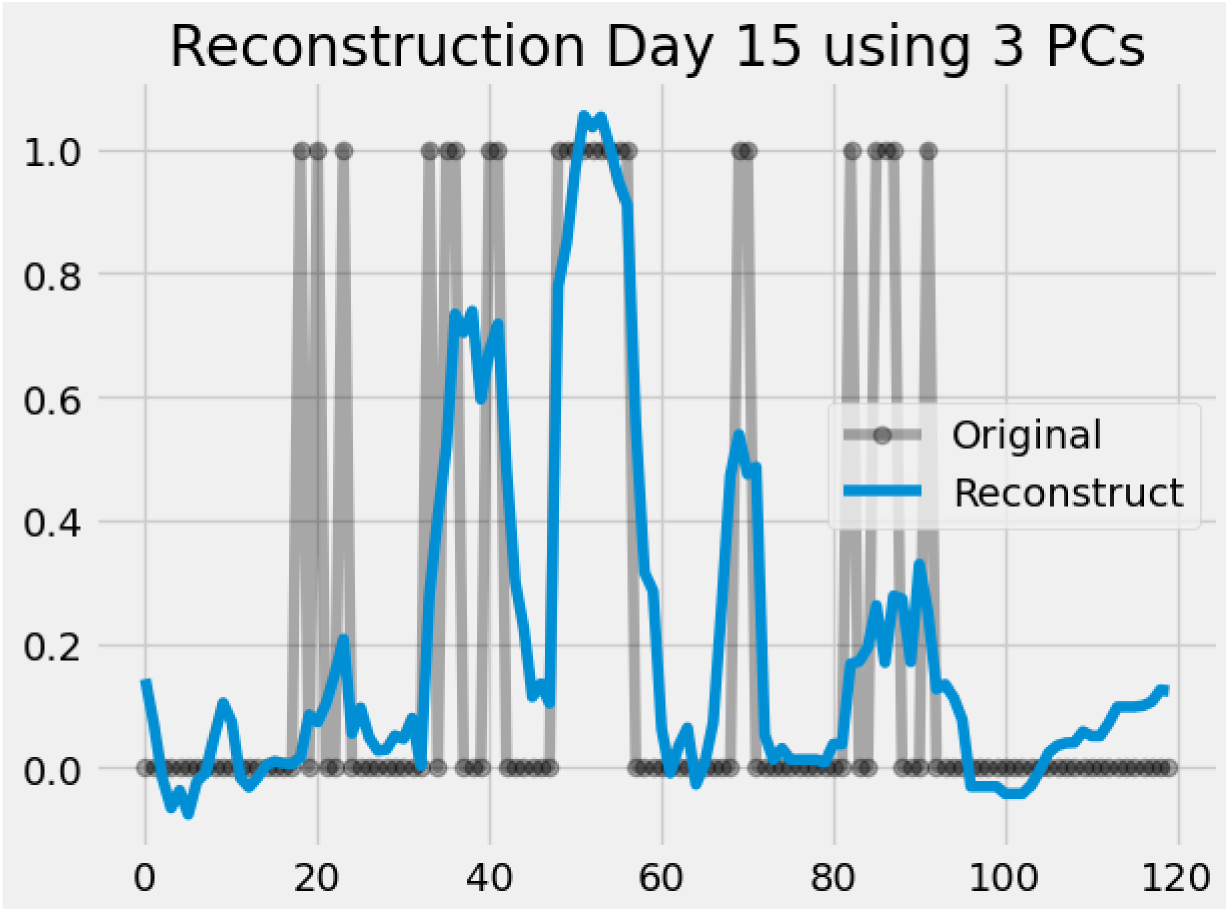


In [38]:

```
# shows the reconstruction of Day 15

x = np.arange(hours*num_labels)
plt.plot(x,Mbw25[14,:], 'o-', color='black', alpha=0.3,label="Original")
plt.plot(x,MBW_recons25_3[14,:],label="Reconstruct")

plt.title('Reconstruction Day 15 using 3 PCs')
plt.tight_layout()
plt.legend()
```

Out[38]:

```
<matplotlib.legend.Legend at 0x40dda8110>
```

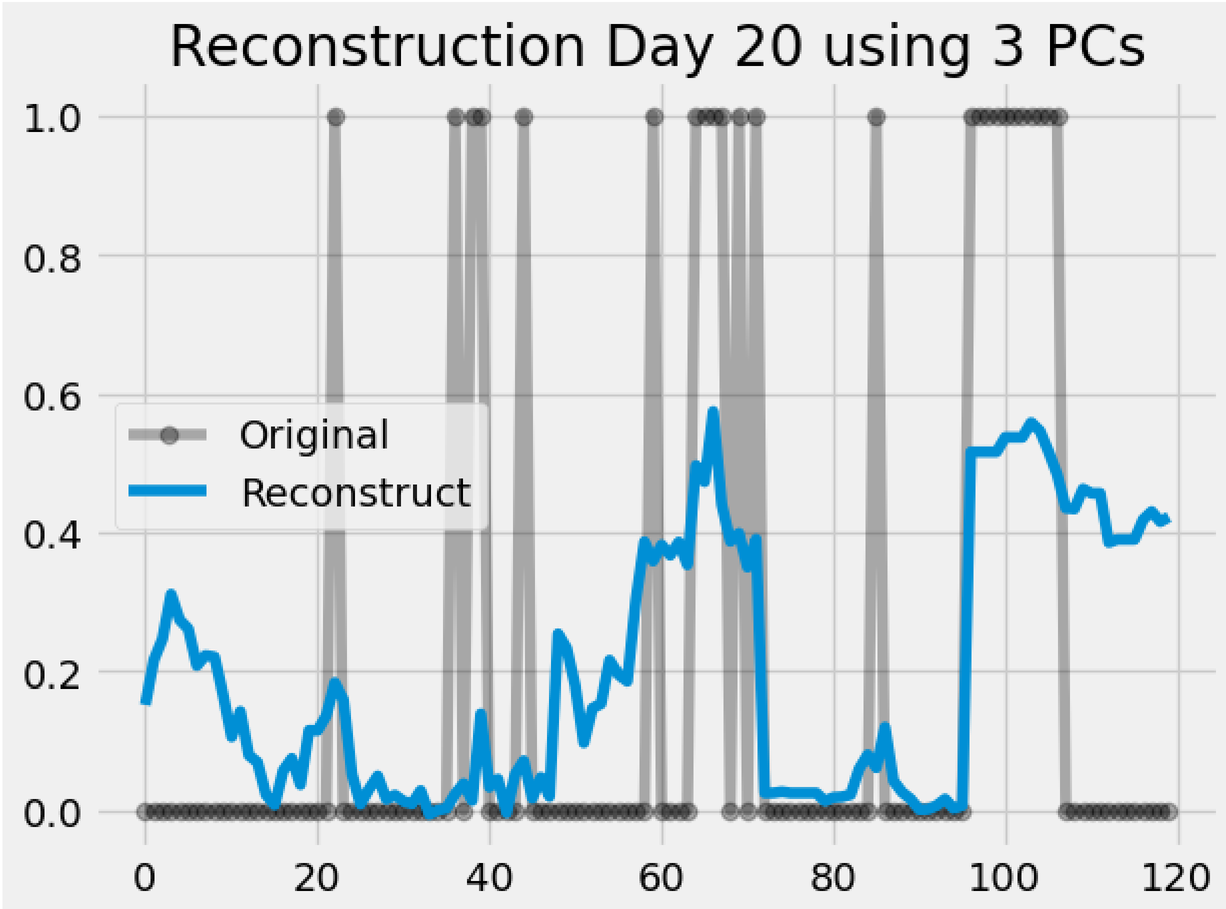


In [39]:

```
# shows the reconstruction of Day 20

x = np.arange(hours*num_labels)
plt.plot(x,Mbw25[19,:], 'o-', color='black', alpha=0.3,label="Original")
plt.plot(x,MBW_recons25_3[19,:],label="Reconstruct")

plt.title('Reconstruction Day 20 using 3 PCs')
plt.tight_layout()
plt.legend()
```

Out[39]:

```
<matplotlib.legend.Legend at 0x347f18a90>
```



In [40]:
```
# Question 3

## Variance attributed to each PC

variance_pc1 = pca25.explained_variance_ratio_[0:119][0]
variance_pc2 = pca25.explained_variance_ratio_[0:119][1]
variance_pc3 = pca25.explained_variance_ratio_[0:119][2]

print(f"The percentage of variance explained by PC1 is {((variance_pc1)*100):.2
print(f"The percentage of variance explained by PC2 is {((variance_pc2)*100):.2
print(f"The percentage of variance explained by PC3 is {((variance_pc3)*100):.2
print(f"Therefore, the total variance explained by the first three components i
```

```
The percentage of variance explained by PC1 is 24.99%
The percentage of variance explained by PC2 is 13.66%
The percentage of variance explained by PC3 is 6.81%
Therefore, the total variance explained by the first three components is 45.4
6%
```

In [41]:
```
# DAY 10
for i in range(1,120):
    pca25_n = PCA(i)
    MBW_recons25_n = pca25_n.inverse_transform(pca25_n.fit_transform(Mbw25))
    acc = 1.00-np.sum(np.square(Mbw25[9,:]-MBW_recons25_n[9,:]))/np.sum(np.squa
    if acc>=0.80:
        print(f"The first {i} PCs achieve a reconstruction accuracy of {acc} fo
        break

# DAY 15
for i in range(1,120):
    pca25_n = PCA(i)
```

```
    MBW_recons25_n = pca25_n.inverse_transform(pca25_n.fit_transform(Mbw25))
    acc = 1.00-np.sum(np.square(Mbw25[14,:]-MBW_recons25_n[14,:]))/np.sum(np.sq
    if acc>=0.80:
        print(f"The first {i} PCs achieve a reconstruction accuracy of {acc} fo
        break

# DAY 20
for i in range(1,120):
    pca25_n = PCA(i)
    MBW_recons25_n = pca25_n.inverse_transform(pca25_n.fit_transform(Mbw25))
    acc = 1.00-np.sum(np.square(Mbw25[19,:]-MBW_recons25_n[19,:]))/np.sum(np.sq
    if acc>=0.80:
        print(f"The first {i} PCs achieve a reconstruction accuracy of {acc} fo
        break
```

```
The first 7 PCs achieve a reconstruction accuracy of 0.8035822443195857 for da
y 10
The first 22 PCs achieve a reconstruction accuracy of 0.8454542417958437 for d
ay 15
The first 8 PCs achieve a reconstruction accuracy of 0.8259303516093788 for da
y 20
```

In [42]:

```
## Q4

accuracy_dict = {}

for i in range(0,((MBW_recons25_3.shape)[0])):
    pca25_3 = PCA(3)
    MBW_recons25_3 = pca25_3.inverse_transform(pca25_3.fit_transform(Mbw25))
    acc = 1.00-np.sum(np.square(Mbw25[i,:]-MBW_recons25_3[i,:]))/np.sum(np.squa
    # Correct for the indexing starting at 0
    day_in_laymens = i + 1
    accuracy_dict[f"Day {day_in_laymens}"] = acc

min_value = min(accuracy_dict.values())
min_key = [key for key, value in accuracy_dict.items() if value == min_value][(

print(f"The day with the minimum accuracy score when using 3 PCs to reconstruct
```

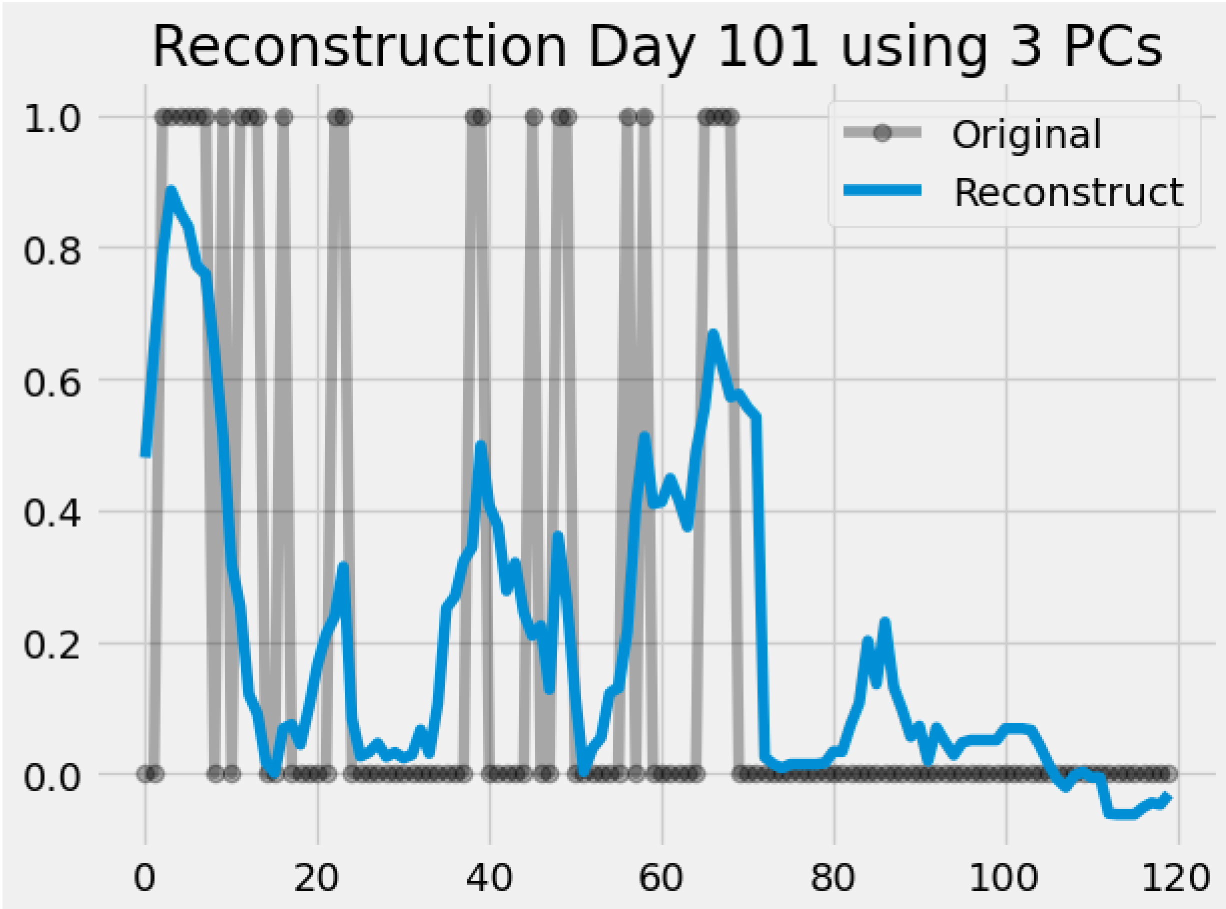
```
The day with the minimum accuracy score when using 3 PCs to reconstruct is: Da
y 101. Because the accuracy is the lowest of all the accuracy scores for each
day when represented by 3 PCs,
it is the worst represented by this number of components.
```

In [43]:

```
# Let's take a look at that reconstruction for the purposes of intuition

x = np.arange(hours*num_labels)
plt.plot(x,Mbw25[101,:], 'o-', color='black', alpha=0.3,label="Original")
plt.plot(x,MBW_recons25_3[101,:],label="Reconstruct")

plt.title('Reconstruction Day 101 using 3 PCs')
plt.tight_layout()
plt.legend()
```

Out[43]:

```
<matplotlib.legend.Legend at 0x3a5324c50>
```



Part 2 - K Means

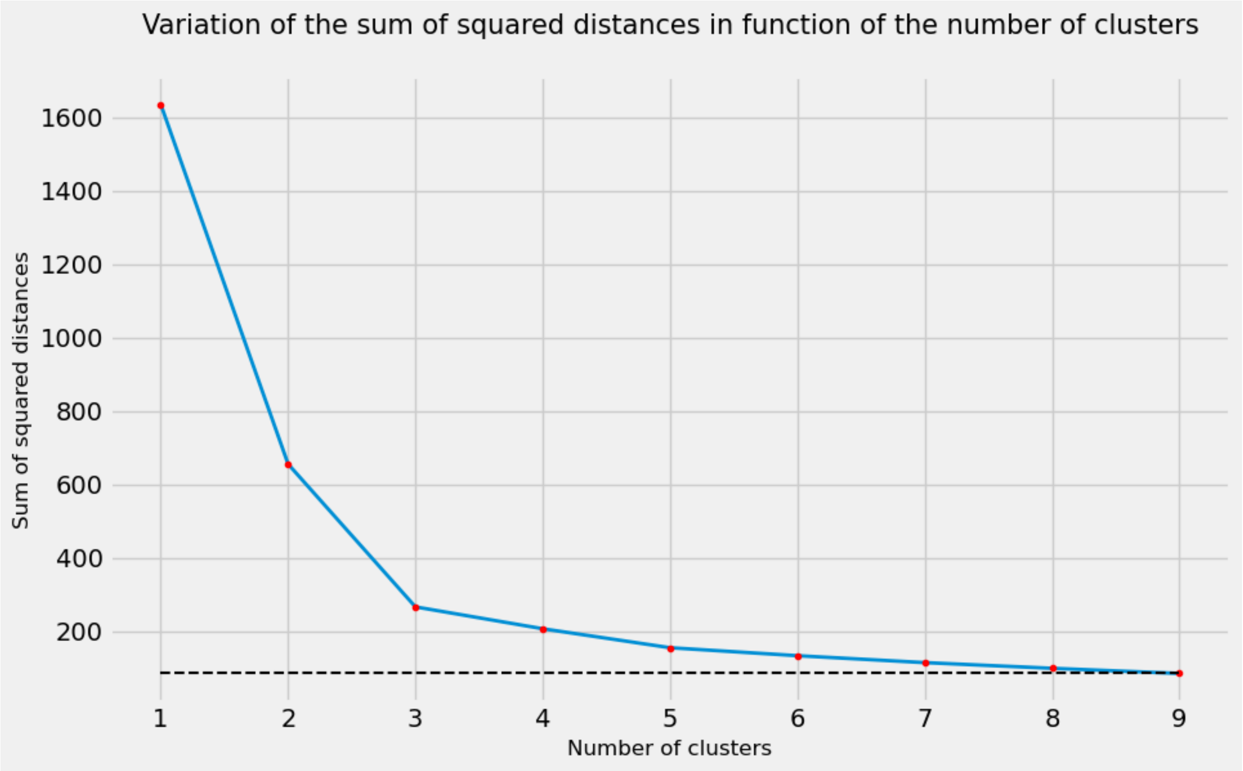
In [44]:
```
Mbw4 = (load_subject_data(4, s))[0]
Mbw16 = (load_subject_data(16, s))[0]
```

In [45]:
```
# 25 - ideal number of clusters is 3

def create_projected_data(n_components, loaded_subject_array):
    pca_n = PCA(n_components)
    projected_data = pca_n.fit_transform(loaded_subject_array)
    return projected_data, pca_n

def elbow_test(projected_data):
    Sum_of_squared_distances = []
    K = range(1,10)
    for k in K:
        km = KMeans(n_clusters=k, n_init=10)
        km = km.fit(projected_data)
        Sum_of_squared_distances.append(km.inertia_)


    plt.figure(figsize = (10,6))
    ax = plt.gca()
    plt.plot(K, Sum_of_squared_distances, linewidth = 2)
    plt.plot(K, Sum_of_squared_distances, '.', c='r',markersize = 6)
    plt.plot(K,[Sum_of_squared_distances[-1] for i in range(len(Sum_of_squared_
            linewidth = 1.5, c = 'black')

    plt.xlabel('Number of clusters', fontsize = 12)
```

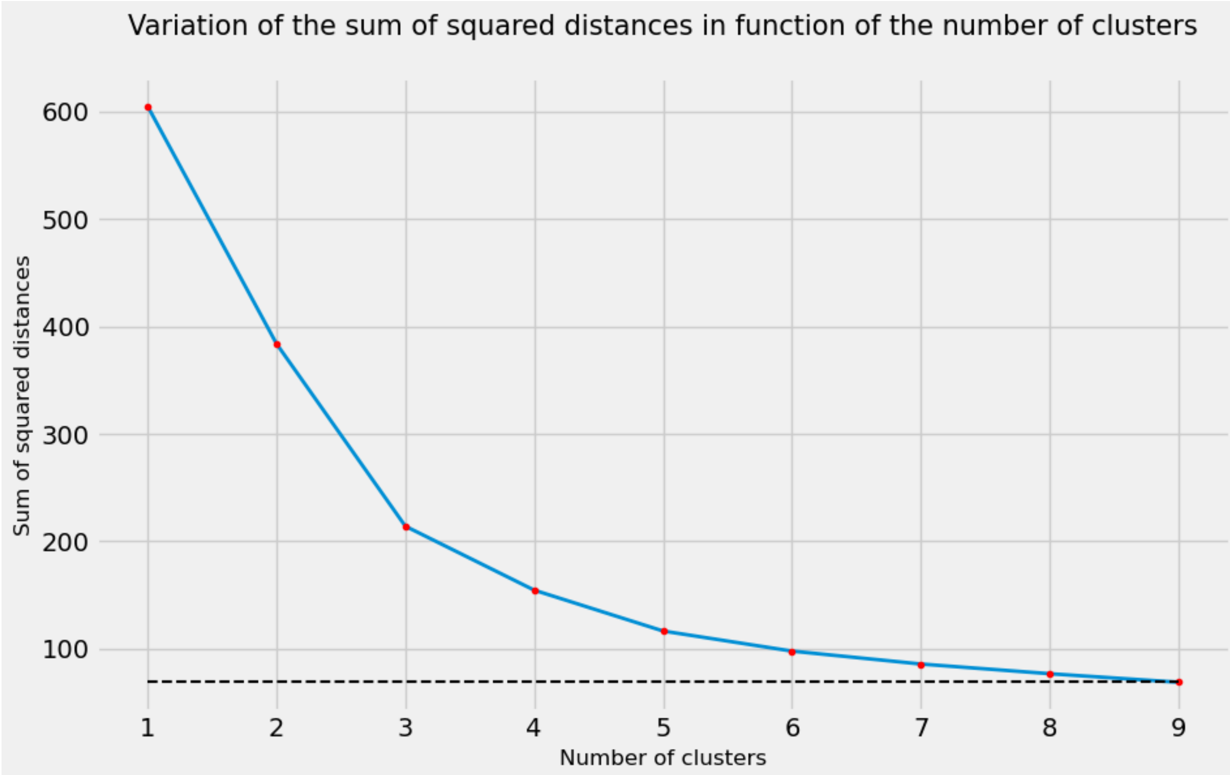
```
    plt.ylabel('Sum of squared distances', fontsize = 12)
    plt.title('Variation of the sum of squared distances in function of the num
    ax.yaxis.set_ticks_position('none')
    ax.xaxis.set_ticks_position('none')
    ax.grid(True)
    return plt.show()
```

In [46]:
```
# Run the elbow test on all subjects -- even as you add more components, it see

# Subject 4
projected_data_4, pca4_3 = create_projected_data(3, Mbw4)
print("The elbow test for subject 4 is:")
elbow_test(projected_data_4)


# Subject 16
projected_data_16, pca16_3 = create_projected_data(3, Mbw16)
print("The elbow test for subject 16 is:")
elbow_test(projected_data_16)


# Subject 25
projected_data_25, pca25_3 = create_projected_data(3, Mbw25)
print("The elbow test for subject 25 is:")
elbow_test(projected_data_25)
```

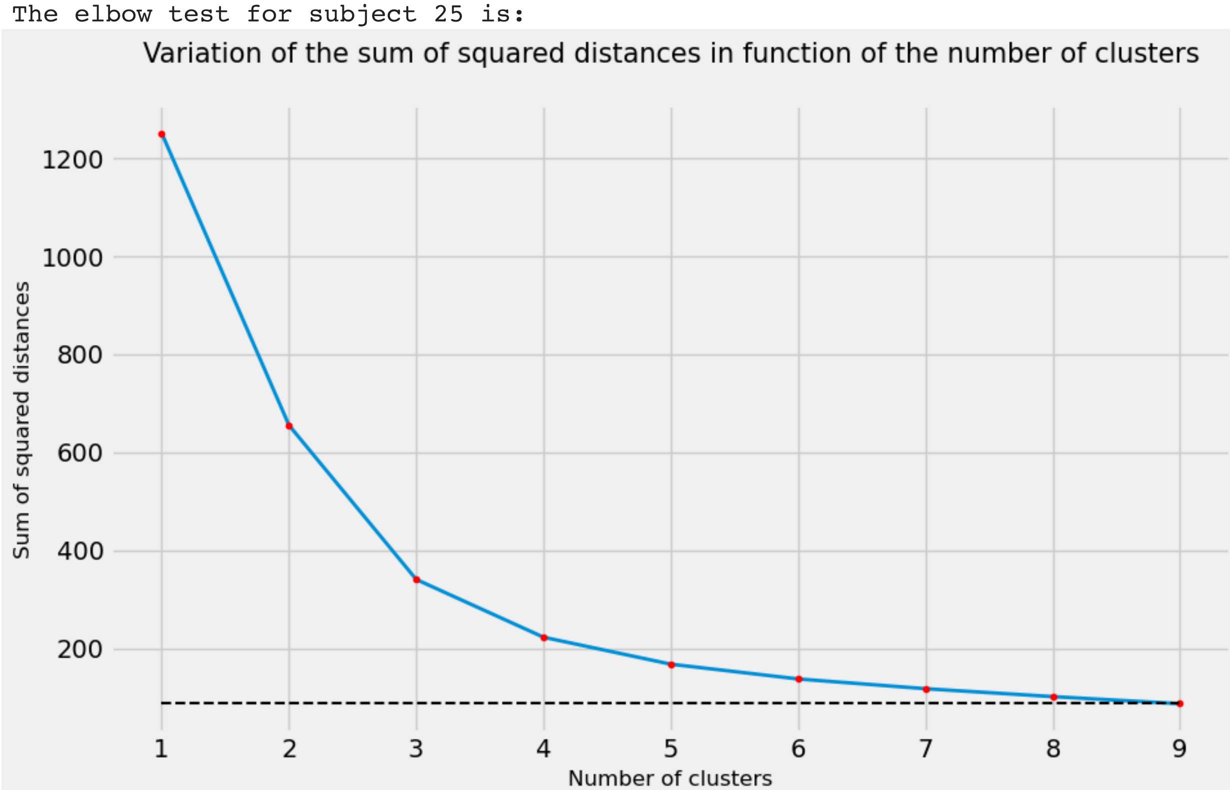
```
The elbow test for subject 4 is:
```




```
The elbow test for subject 16 is:
```




```
The elbow test for subject 25 is:
```

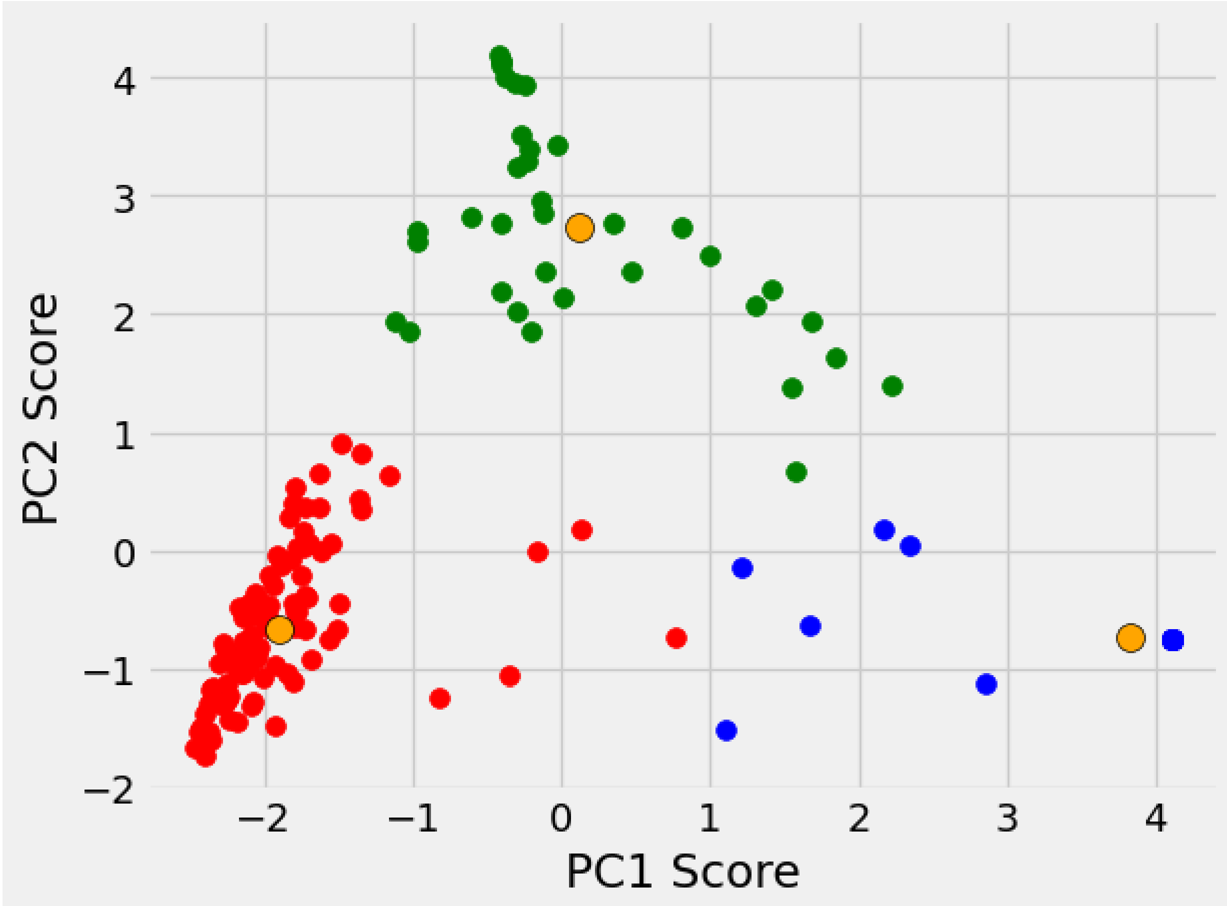


Based on the above information - 3 clusters seems to be the ideal number of clusters

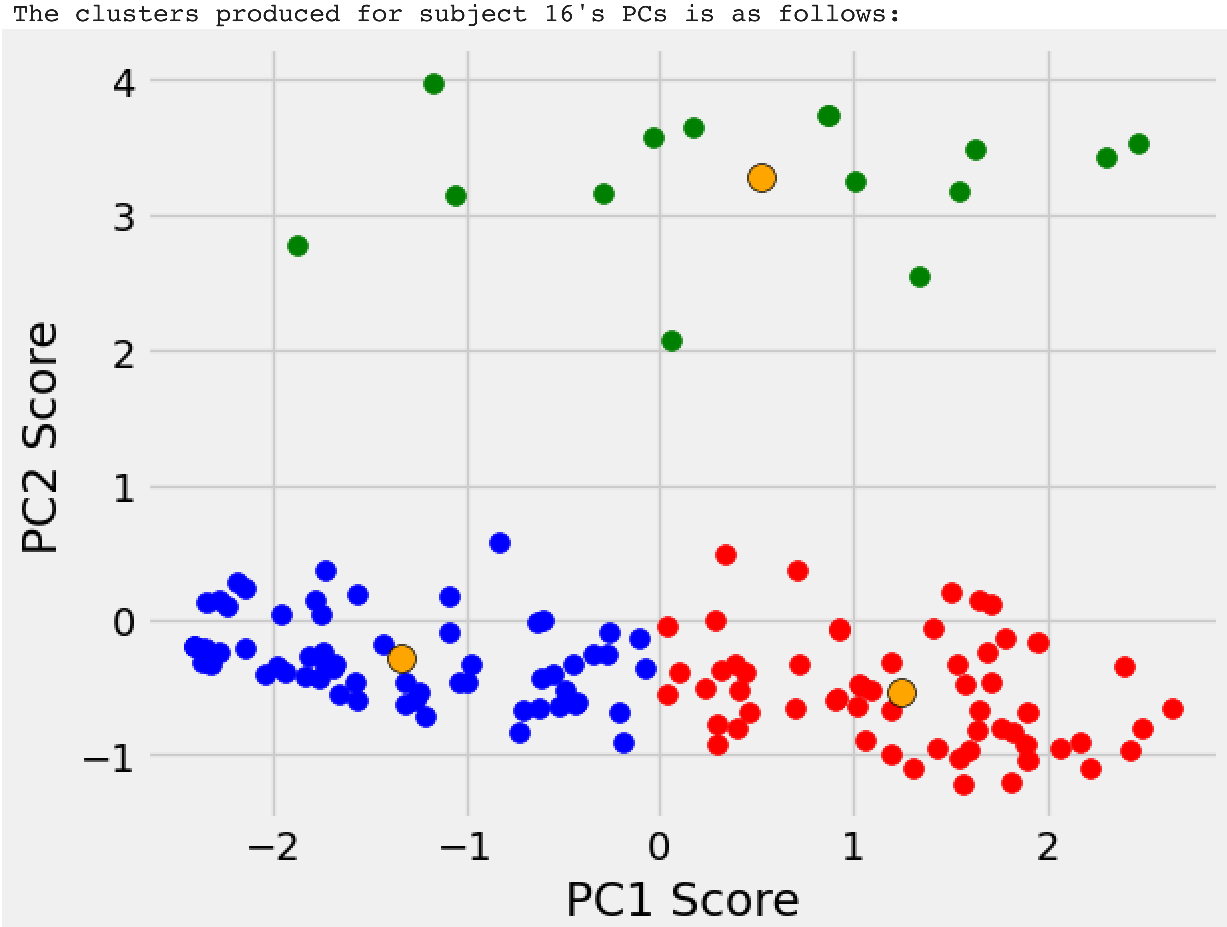
In [68]:
```
def run_plot_kmeans(n_clusters, projected_data, pca_n):

    kmeans = KMeans(n_clusters = n_clusters,random_state=1, n_init = 10)
    membership = kmeans.fit_predict(projected_data) #Important this can be done
    Score = kmeans.score(projected_data)
    centers = kmeans.cluster_centers_
    centers_initial_base = pca_n.inverse_transform(centers)
```

```
    inertia = kmeans.inertia_
    y_km = membership

    colors = ['red', 'blue', 'green', 'orange', 'purple', 'pink']
    f = 0
    for i in range(0, n_clusters):
        plt.scatter(projected_data[y_km ==i,0], projected_data[y_km == i,1], s=
        f+=1

    plt.scatter(kmeans.cluster_centers_[:, 0],kmeans.cluster_centers_[:, 1], c=

    plt.xlabel("PC1 Score")
    plt.ylabel("PC2 Score")
    plt.tight_layout()

    plt.show()

    return membership, inertia, kmeans.cluster_centers_

# Subject 4
print("The clusters produced for subject 4's PCs is as follows: ")
s4_km_membership, s4_inertia, s4_centroids = run_plot_kmeans(3, projected_data_


# Subject 16
print("The clusters produced for subject 16's PCs is as follows: ")
s16_km_membership, s16_inertia, s16_centroids = run_plot_kmeans(3, projected_da


# Subject 25
print("The clusters produced for subject 25's PCs is as follows: ")
s25_km_membership, s25_inertia, s25_centroids = run_plot_kmeans(3, projected_da
```

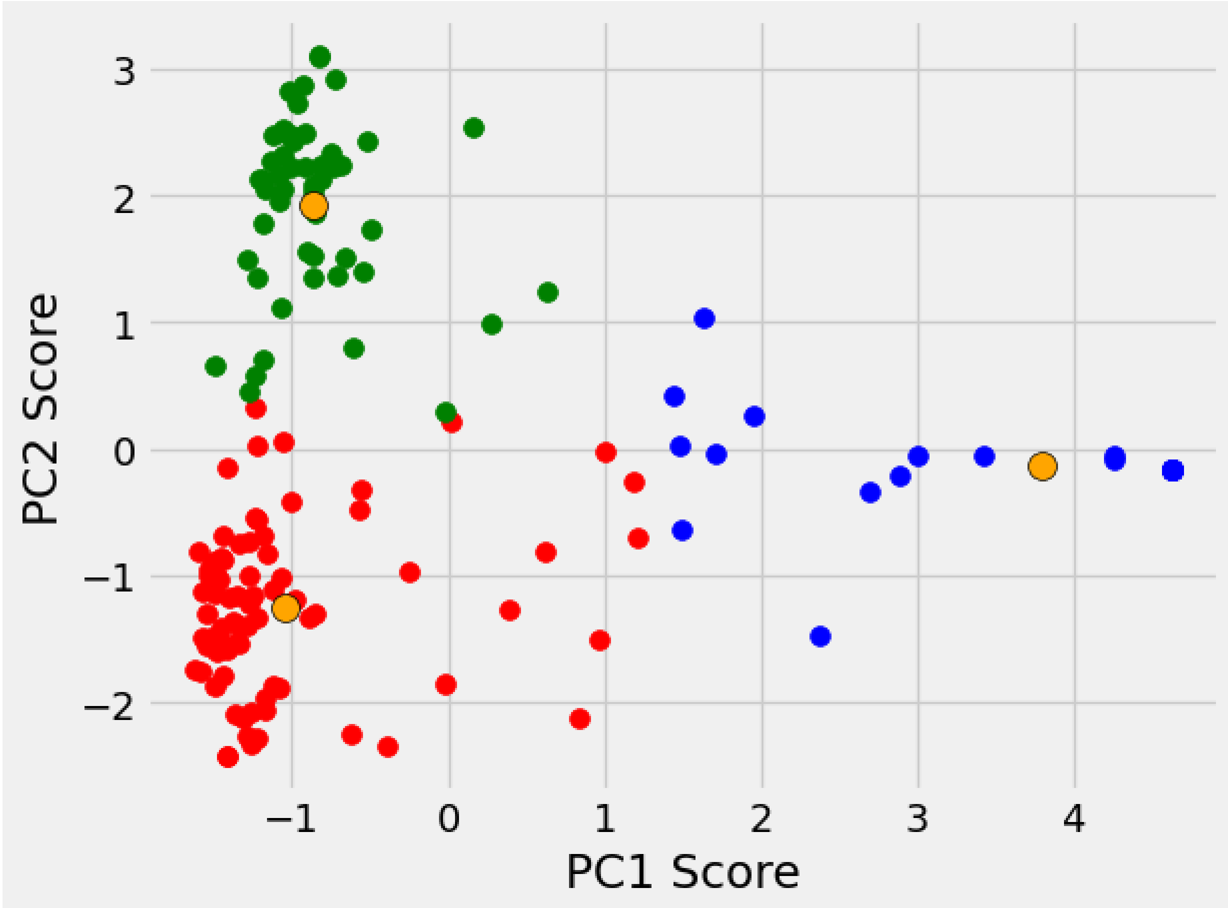
```
The clusters produced for subject 4's PCs is as follows:
```




```
The clusters produced for subject 16's PCs is as follows:
```




```
The clusters produced for subject 25's PCs is as follows:
```



It seems as though subject 4's behavior aligns more with the behavior of subject 25 just based on a visual inspection of the chart. The red and green clusters seem to be clustered densely on the lefthand side of the chart, whereas the blue cluster is sparsely populated and sits on the lefthand side of the plot. Additionally, based on the analysis of membership below, the distribution of PCs in each cluster for subject 4 is far more similar to that of subject 25 than that of subject 16.

In [84]:
```
euclidean_distances_s4_s16 = [euclidean(p1, p2) for p1, p2 in zip(s4_centroids,
euclidean_distances_s4_s25 = [euclidean(p1, p2) for p1, p2 in zip(s4_centroids,
```

In [89]:
```
print("The individual euclidean distances for the centroids of clusters in s4 v
print(euclidean_distances_s4_s16)
print(np.mean(euclidean_distances_s4_s16))
print("\n")
print("The individual euclidean distances for the centroids of clusters in s4 v
print(euclidean_distances_s4_s25)
print(np.mean(euclidean_distances_s4_s25))
```

```
The individual euclidean distances for the centroids of clusters in s4 vs thos
e in s16 are:
[3.1443840578238293, 5.17036901728634, 0.7282249895075766]
3.014326021539249


The individual euclidean distances for the centroids of clusters in s4 vs thos
e in s25 are:
[1.0378568141296352, 0.6022257544088024, 1.2727548713251997]
0.9709458132878792
```

Given the graphs and the quantitative analysis of the distances between centroids of subject 4's clusters vs both subject 16 and 25's clusters, it is safe to say that subject 4 is far more similar in behavior patterns to subject 25. We can see the distribution of data amongst clusters is also more similar between s4 and s25 based on the below code.

In [52]:
```
def membership_stats(km_membership, n_clusters, subject_number):
    print(f"The k-means membership statistics for subject {subject_number} are:
    N = len(km_membership)
    percentage_list = []
    nb_of_people_list = []
    for i in range(n_clusters):
        percentage_list.append(round(100*(km_membership== i).sum()/N,2))
        nb_of_people_list.append((km_membership == i).sum())
        print("The cluster " + str(i + 1) + " includes {:.2f}%".format(percenta
    print("\n")

# Subject 4
membership_stats(s4_km_membership, 3, 4)
print(f"The inertia for clustering on subject 4 is {s4_inertia}\n")

# Subject 16
membership_stats(s16_km_membership, 3, 16)
print(f"The inertia for clustering on subject 16 is {s16_inertia}\n")

# Subject 25
membership_stats(s25_km_membership, 3, 25)
print(f"The inertia for clustering on subject 25 is {s25_inertia}\n")
```

```
The k-means membership statistics for subject 4 are:
The cluster 1 includes 53.89% of the days.
The cluster 2 includes 26.11% of the days.
The cluster 3 includes 20.00% of the days.


The inertia for clustering on subject 4 is 267.1676058659599

The k-means membership statistics for subject 16 are:
The cluster 1 includes 43.70% of the days.
The cluster 2 includes 45.19% of the days.
The cluster 3 includes 11.11% of the days.


The inertia for clustering on subject 16 is 214.18011019768733

The k-means membership statistics for subject 25 are:
The cluster 1 includes 47.53% of the days.
The cluster 2 includes 20.37% of the days.
The cluster 3 includes 32.10% of the days.


The inertia for clustering on subject 25 is 340.6013714184267
```

In [ ]: